Characterization of model-based reasoning strategies for use in IVHM architectures

Scott Poll, David Iverson, Ann Patterson-Hine
NASA Ames Research Center, Computational Sciences Division, Moffett Field, CA, USA 94035

ABSTRACT

Open architectures are gaining popularity for Integrated Vehicle Health Management (IVHM) applications due to the diversity of subsystem health monitoring strategies in use and the need to integrate a variety of techniques at the system health management level. The basic concept of an open architecture suggests that whatever monitoring or reasoning strategy a subsystem wishes to deploy, the system architecture will support the needs of that subsystem and will be capable of transmitting subsystem health status across subsystem boundaries and up to the system level for system-wide fault identification and diagnosis. There is a need to understand the capabilities of various reasoning engines and how they, coupled with intelligent monitoring techniques, can support fault detection and system level fault management. Researchers in IVHM at NASA Ames Research Center are supporting the development of an IVHM system for liquefying-fuel hybrid rockets. In the initial stage of this project, a few readily available reasoning engines were studied to assess candidate technologies for application in next generation launch systems. Three tools representing the spectrum of model-based reasoning approaches, from a quantitative simulation based approach to a graph-based fault propagation technique, were applied to model the behavior of the Hybrid Combustion Facility testbed at Ames. This paper summarizes the characterization of the modeling process for each of the techniques.

Keywords: Model-based reasoning, diagnostic reasoning, fault diagnosis, hybrid rockets

1. INTRODUCTION

A Hybrid Combustion Facility (HCF) was recently built at NASA Ames Research Center to investigate the combustion properties of a new fuel formulation developed by Stanford University researchers. A hybrid rocket is one in which the fuel is in solid form and the oxidizer is in liquid or gaseous form. The fuel being tested at the HCF is paraffin-based, similar to candle wax, and the oxidizer is gaseous oxygen.

The primary advantage of hybrid rockets over liquid and solid rockets is the inherently safe nature of the fuel—in manufacturing, handling, and operationally. The fuel by itself is not volatile, which leads to a number of cost reductions of a vehicle launch system. In addition, the products of combustion are harmless carbon dioxide and water. Unlike solid rockets, hybrid rockets can be throttled to change the thrust after they are ignited. Because of the dramatically improved performance of the paraffin-based fuel compared to previous hybrid fuels, hybrid rockets now have the potential to be safer, less expensive replacements to the solid rocket boosters on the space shuttle or other launch systems.

The HCF was constructed to see if promising results of bench-top experiments would scale up to a size that is closer to an operational rocket and to examine the physical mechanisms of the liquid-layer combustion process. Several firings of the facility have been completed and more are planned in the future.

Researchers at Ames working in the area of Integrated Vehicle Health Management (IVHM) recognized a good opportunity to apply IVHM techniques and concepts to a candidate technology for next generation launch systems. One of the most difficult and time-consuming aspects of putting together an IVHM system is knowledge acquisition. Here, that problem is mitigated because of easy access to the experts (and their willingness to share their knowledge) and the facility. This gives us a chance to examine a variety of IVHM tools, compare and contrast their approaches, and assess the feasibility of using such techniques in a hybrid rocket health management system.

2. HCF OVERVIEW

A detailed description of the Hybrid Combustion Facility can be found in the literature[1]. A brief overview is presented here to give the reader the necessary background to understand the facility operation. Figure 1 and Figure 2 show a composite picture and sketch of the facility, respectively[1]. There are six main systems: liquid oxygen (LOX) feed, gaseous oxygen (GOX) feed, combustion, ignition, pneumatics, and controller.

Prior to a firing, the oxidizer stored in the LOX tank is pumped through the vaporizer and gasified before entering the GOX tank. Over a period of up to an hour, GOX flows into the GOX tank until the pressure reaches the required level for the desired mass flow rate and run duration. At this point, the LOX feed system is isolated from the GOX tank by closing a shut off valve between the vaporizer and the GOX tank.

The operator enters the desired run setpoints into the control computer. These include parameters for control valve scheduling, ignition timing, the desired delivery pressure, and configuration information. After a firing countdown, the upstream shutoff valve, POV-4, is opened (the redundant downstream valve has previously been opened). The GOX flow chokes at the sonic nozzle (orifice) and continues into the combustion chamber. A short time later, the ignition system oxidizer and fuel flow are turned on and ignited by a spark. High temperature combustion products from the ignition system are injected to the combustion chamber and vaporize the paraffin fuel, which mixes with the free stream oxidizer and the ignition products to ignite the paraffin in a self-sustaining combustion reaction.

As the GOX tank pressure decreases during the course of a firing, the control valve opens to maintain constant delivery pressure (and mass flow) to the combustion chamber. A venturi in the GOX feed line measures the oxygen mass flow rate but is accurate only for steady state operating conditions. The sonic nozzle measures the mass flow rate more accurately during transients and also serves to isolate any pressure fluctuations in the combustion chamber from the feed system.

A check valve upstream of the orifice prevents reverse flow of combustion gases from entering the GOX feed line. Two burst disks located just downstream of the orifice and one located upstream of the GOX tank protect against over pressurization.

Pressure sensors are located at the GOX tank, sonic nozzle, and combustion chamber. There is also a high frequency pressure sensor measurement of the combustion chamber pressure and a differential pressure measurement at the venturi. The GOX temperature is measured upstream of the sonic nozzle. All pneumatically actuated valves report open/close status feedback and the burst disks indicate burst/not burst status.

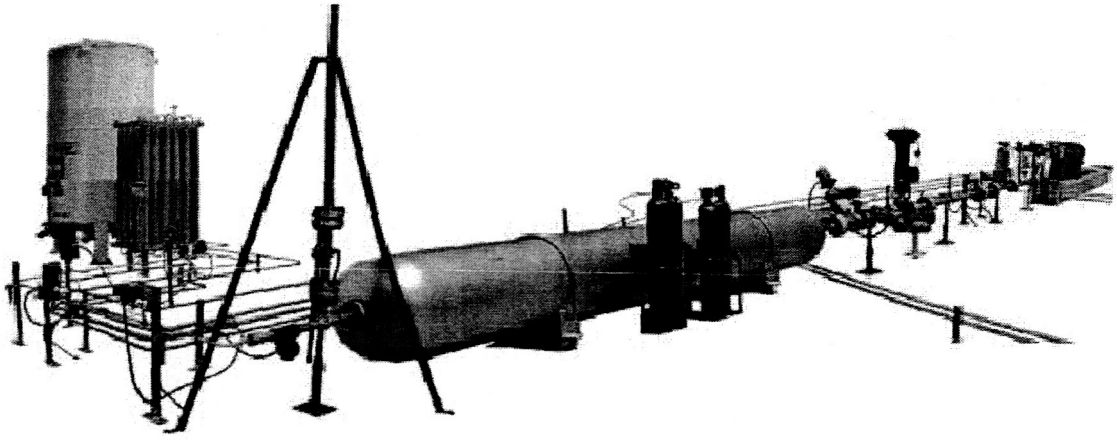

Figure 1: Ames hybrid combustion facility.

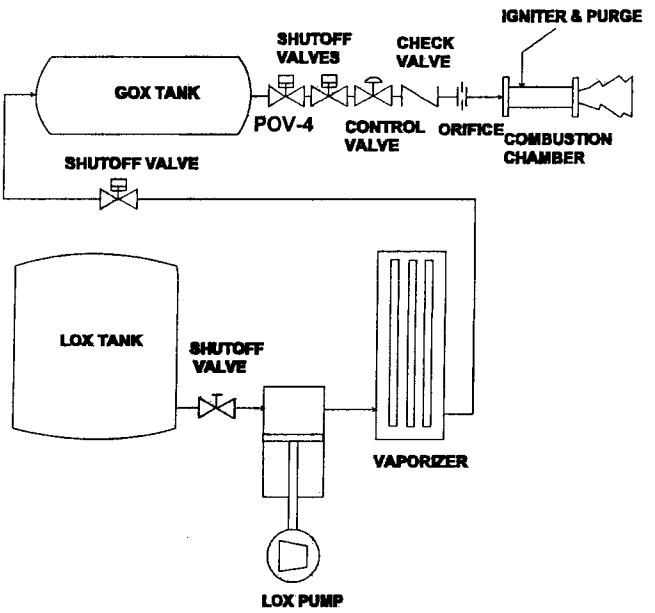

Figure 2: Schematic of hybrid combustion facility.

3. MODEL-BASED REASONING TOOLS

During a firing of the HCF we wish to know whether any of the observed parameters deviate from usual or nominal values. This is fault detection. We may also want to determine the kind of fault, the location and behavior of the fault, and the time at which the fault occurred. This is fault diagnosis. The HCF controller monitors individual signals and reports a fault when a parameter exceeds a threshold or indicates an unacceptable value (for example, if a burst disk sensor indicates that the burst disk has ruptured). Depending on the severity of the fault, the controller will perform an emergency shutdown to safe the facility. Since individual sensors are used for the fault detection, no information about the structure of the system or the relationships among the different sensor readings is captured. A single physical fault may result in many fault indications and it is up to the operator to sort through them to find the common cause and isolate the failure. Alternatively, one could attribute certain combinations of sensor indications to an underlying cause if the operator has built up enough experience with the facility to know how the faults are manifested. Since experience is a key factor, this rule-based approach may not cover certain faults that have not yet been observed and the rules may become invalid if the facility is modified. To overcome these difficulties, fault management systems have increasingly made use of model-based reasoning systems in which the physical structure and behavior of the plant are captured in hierarchical, compositional models. If the observations of the plant deviate from what is expected, a fault is detected and various algorithms may be used to isolate the fault. Figure 3 illustrates the basic idea of model-based reasoning. While there are many model-based reasoning tools, only a few are examined in this study. It should be emphasized that the tools were selected purely on their availability and familiarity to the researchers; there was no intensive survey performed to find which techniques might be best for the task at hand. Rather, the goal was to see how to tackle the problem with the available tools and compare and contrast their approaches. The three tools that are discussed in this paper are TEAMS (Testability Engineering and Maintenance System), L2 (Livingstone2), and RODON.

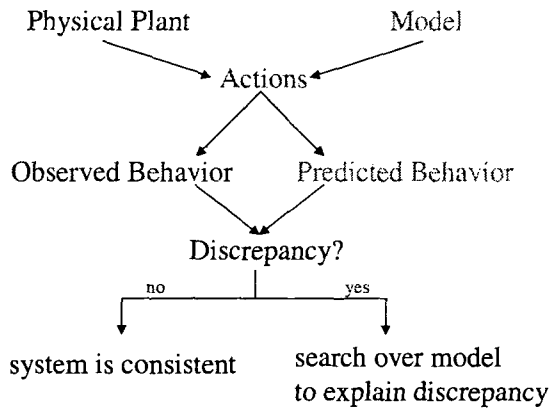

Figure 3: Basic idea of model-based reasoning.

3.1 TEAMS

One method selected for study is the causal model-based technique implemented in a commercial tool set from Qualtech Systems, Inc. Qualtech's integrated tool set for design-for-testability, interactive trouble-shooting and on-line monitoring and diagnostics, includes the three tools developed using NASA Small Business Innovation Research funding: TEAMS, TEAMS-RT, TEAMS-RDS. These tools are founded on the multi-signal flow graph modeling methodology and the concomitant fault-isolation algorithms. TEAMS integrates the modeling methodology and fault-isolation algorithms in an easy-to-use graphical user interface. TEAMS is mainly a design-for-testability tool, but the same diagnostic model is used with its companion tool, TEAMS-RT (a real-time diagnostics tool) to perform passive on-line fault-diagnosis using asynchronously arriving anomaly reports, alarms, or applied test-results. The TEAMS-RDS (Remote Diagnostic Server) product incorporates TEAMS-RT and other Qualtech software components on a server computer that can "serve" intelligent, optimized diagnostics to thin clients over the Internet or any computer network. All diagnostic reasoning and technical data can be maintained and upgraded on a central server and instantly made available to clients on the system (via local area network or Internet).

The multi-signal modeling methodology is a hierarchical modeling methodology where the propagation paths of the effects of a failure are captured in terms of a directed graph[4]. The model is developed by entering the structure of the model, based on a schematic diagram or conceptual block diagram, and then adding signals to the modules as well as test points. Signals describe the unique attributes of the variables in a system. Test points designate locations of visibility into the system. For example, physical locations of sensors such as pressure transducers would have a corresponding test point on the multi-signal flow graph. Multiple tests can be defined at a given test point. Test results can come from simple limit checks, feature extractions using signal analysis techniques, more complex data analysis algorithms, or even other diagnostic reasoners. Propagation algorithms convert this graph to a single global fault dictionary for a given mode of the system. This dictionary contains the basic information needed to interpret test results and diagnose failures during on-board monitoring. Multi-signal modeling allows the modeler to hierarchically describe the structure of a system and then specify its functional attributes via signals. It is not a simulation model but is ideally suited for building accurate low-cost models that can be used by a reasoner in real-time to interpret test results and assess system health.

An important aspect of multi-signal modeling is identification of signals—a process in which the modeler summarizes his understanding of the functions of components in the system in terms of their distinct attributes. Tests are procedures that look at the data from the sensors and make decisions about system attributes associated with those measurements. The test definition can include additional information such as test cost, test time (time required to perform the test), detection probability, false alarm probability, as well as a test label. Some of the test parameters are used by the TEAMS algorithm to optimize a troubleshooting tree. Test labels are useful for assessing the diagnosability of the system using various levels of instrumentation. The detection and isolation coverage available with a particular instrumentation configuration is determined by performing a testability analysis. The testability analysis is done on the same model that is then used by the QSI's real-time reasoning software. Using a consistent model during the design

phase through operations enables continuous verification of the models by system experts and increased confidence in the automated system.

In this application, the testability analysis can contribute to advanced sensor development efforts by suggesting optimal sensor placement and analyzing possible redundancies in sensor coverage. The fault detection and isolation coverage analyses can aid the implementation of the real-time reasoner, whether it is the companion tool, TEAMS-RT, or some other method.

3.2 Livingstone 2

Livingstone is an open-source model-based reasoning tool that was developed at NASA Ames in the past decade and recently enhanced to L2. Some relevant papers are Williams and Nayak[5] and Kurien and Nayak[6]. Livingstone was one of the component technologies demonstrated in the Deep Space-1 Remote Agent Experiment[7].

The fundamental tasks of Livingstone are to eavesdrop on system commands, to determine whether those commands had the desired effect on the spacecraft or plant, and to recommend reconfiguration actions to achieve a desired goal in the event of failures. To accomplish these tasks, a declarative model of the system is built in a hierarchical, compositional framework. Automata (components) describe system behavior with a finite number of nominal and failure modes and by transitions between the modes. Within each mode, propositional formulae relate the component inputs to outputs and define the functional behavior of the component in that particular mode. The inputs and outputs may be connected to other components in the model. Transitions between nominal modes represent commanded configuration changes while transitions from nominal to failure modes are unexpected failure events. The nominal transitions have an associated cost while the failure transitions have an associated prior probability of occurrence. Each automaton has a transition variable with the nominal and failure transitions in its domain. The set of assignments to each transition variable at each time step describes a trajectory of system evolution. L2 incrementally generates multiple trajectories, in order of likelihood, that are consistent with the commands and observations.

Given the commands to the system, L2 attempts to transition the model to the next nominal configuration. If the observed sensor values violate the constraints imposed by the propositional formulae of the intended nominal modes, the conflict results in a fault being detected, and L2 diagnoses what the failed system state is. L2 uses a conflict-directed best-first search to efficiently find consistent candidate trajectories. Conflict-directed refers to using the conflict (discrepancy) to avoid generating candidates that contain assignments to the variables that would include the conflict. Best-first search refers to checking the consistency of the most likely candidates first. If none of the candidates of a certain likelihood are consistent, the next most probable candidates are checked for consistency and so on. Diagnosis amounts to finding the most likely assignments to the transition variables that are consistent with the commands and observations. Recovery, or reconfiguration, addresses finding the least costly actions (transitions) required to move the system into the desired state. The recovery feature of L2 was not used in this study.

A Livingstone model is a discrete representation of a physical system. This means that real-valued sensor data and system behavior must be abstracted into a discrete space. Discrete variable values typically represent a range of real-valued numbers or a range of the rate of change of real-valued numbers and are chosen to aid the diagnosis of the system. Monitor code external to L2 performs the discretization of the real-valued sensor data to the discrete variable values used in L2.

3.3 RODON

RODON is a commercial model-based reasoning software tool produced by ROSE Informatik. It uses a mathematical system description for simulation, performance analysis, risk analysis, monitoring, and diagnosis of complex technical systems. The behavior of each system component is modeled using familiar engineering equations combined with logic clauses. The topology of the system is modeled by connecting these components together via input and output ports. These models are entered into RODON using a graphical system editor and component editor.

In contrast to other diagnostic tools used on this project, RODON performs numerical calculations and reasons directly with system sensor values rather than abstracting these values into discrete bins or working with results of external tests. This allows RODON to reason with higher fidelity system models that can provide tighter monitoring tolerances and, in

many cases, more accurate diagnoses. One potential draw back of this detailed modeling capability is an increase in computational complexity that can lead to a slower diagnostic response time than we might experience with the other, more abstract, diagnostic tools. The response time issue can be addressed by the modeler to a certain extent by removing unnecessary detail from the model. System sensor placement, type, and tolerance as well as the desired diagnostic response time can help the modeler determine an appropriate level of model detail.

The basis of RODON's reasoning is a constraint satisfaction algorithm that determines if the data provided by the system sensors can be used to derive a consistent instantiation of the mathematical model. In other words, can the sensor readings be propagated through the model in such a way that a set of equations describing nominal system behavior all hold true? RODON performs system monitoring by collecting system sensor data and executing this consistency check. If the sensor values lead to a consistent model solution, RODON determines that the system is functioning properly. If no consistent solution can be found, RODON can use the same system model to perform a diagnosis. Diagnoses are accomplished by suspending the constraints imposed by the equations describing each component and attempting to solve the resulting model with the provided sensor data. If a consistent solution can be found when the constraints for a particular component are suspended, that component is presented as a suspected failure. Although the underlying algorithm is more complex and efficient, the end effect is to suspend nominal behavior for each component, one by one, and check for model consistency. While not required, RODON also permits the modeler to mathematically describe failure modes of each component. If, during a diagnosis, a consistent solution can be found by substituting the failure mode constraints for a component's nominal constraints, then that failure mode will be presented as a suspected failure.

One other unique feature of RODON is the use of interval arithmetic for its calculations. Rather than requiring that each variable used in an equation be assigned a single numeric value, it can be assigned ranges of values or intervals. These intervals will be used in the solution of all the equations containing that variable. If certain values in the interval would cause one of the equations containing the variable to become invalid, those values are removed from the interval. If there are no values in the interval that could be used to make the equation valid, then a conflict occurs and that particular equation is no longer allowed for consideration in the model solution. The use of intervals in RODON provides a convenient method for incorporating uncertainty, such as component or sensor tolerances, into the system model.

4. HCF MODELS

Models of the HCF were created with TEAMS, L2, and RODON. The commands and observations that were input to the models were taken from logged data files; real-time systems were not developed. The scope of the TEAMS model included all of the subsystems; the scope of the L2 model included the GOX feed and combustion chamber subsystems; the scope of the RODON model included the GOX feed subsystem only. Figure 4 shows the top level of the L2 model. A detailed description of the models is not possible here due to restrictions on paper length. The full details of the models will be published in a NASA technical memorandum. In order to illustrate the modeling process with the three tools we will focus the discussion on the main shut-off valve, POV-4. A picture of the physical valve assembly is shown in Figure 5. An electrically actuated solenoid valve is plumbed to a supply of compressed air at the inlet side. The outlet side of the solenoid valve is connected to the actuator. After the valve is commanded open, the compressed air is allowed to pass through the solenoid valve to the actuator, where it drives a piston that rotates the ball valve to the open position. Switches in the enclosure on top of the valve will close or open depending on the orientation of the valve stem.

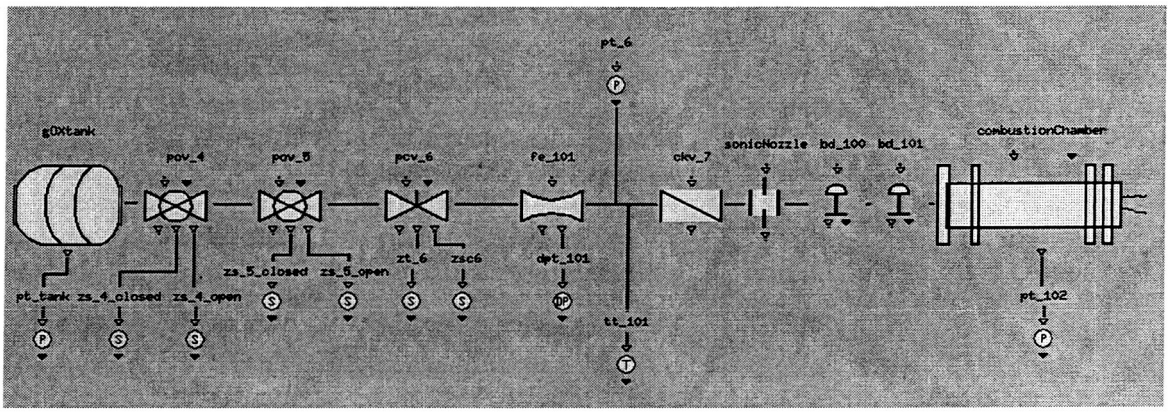

Figure 4: Livingstone HCF model.

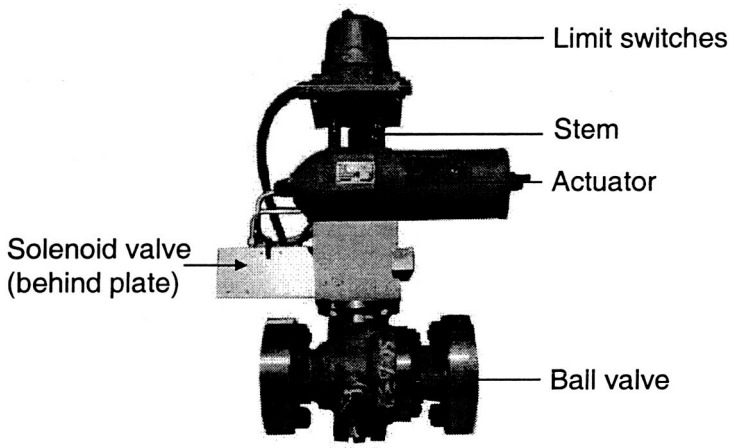

Figure 5: Valve POV-4.

4.1 TEAMS

Table 1 lists the components, tests, and signals of the valve module. Code external to the model determines when a TEAMS test is applicable. For example, after an open command has been sent to the valve, the tests labeled POV4C_open_cmd and POV4O_open_cmd would be set active and report a pass/fail test result to the real-time diagnostic engine whereas the tests labeled POV4C_close_cmd and POV4O_close_cmd would be inactive and would not report any test results. In this case, the active tests would verify whether or not the open command had the expected effect on the close and open limit switches—i.e., the open limit switch should report "open" and the close limit switch should report "notClosed". If the valve did not open the tests will fail and at least one of the signals attached to each test is implicated as the cause. The signal pneumatic4 is attached to the open_cmd tests and to the components that could prevent the valve from opening, thus causing the tests to fail. This includes the components in the valve as well as the pneumatic subsystem (via the signal mapping, ➔), since the valves are pneumatically actuated.

When the valves are commanded closed, the signals attached to the close_cmd tests are attached only to the valve components since pneumatic subsystem failures will not cause the normally closed valves to fail open. We allow for the case of a valve being stuck in an intermediate position by attaching a signal to the appropriate valve failure mode and limit switch tests; when the valve is commanded open, the test on the close limit switch (POV4C_open_cmd) will pass since it will report "notClosed" but the test on the open limit switch (POV4O_open_cmd) will fail since it will report "notOpen". Similarly, when the valve is commanded closed, the test on the open limit switch (POV4O_close_cmd) will pass since it will report "notOpen" but the test on the close limit switch (POV4C_close_cmd) will fail since it will report "notClosed". If we attach a unique signal (inter4) to the open limit switch test for the open command, to the close limit

switch test for the close command, and to the valve failure mode POV4_intermediate, the valve will be implicated if the limit switches indicate "notOpen" and "notClosed" following a close or open command.

Another valve failure mode is included for a leaky valve. In this case, the limit switches will report the correct valve position but downstream tests will detect if there is an unexpected flow or pressure. Additionally, a faulty limit switch can be implicated if, for example, the limit switch reports closed but a downstream test on pressure or mass flow shows that the valve is in fact open.

Component:failure mode	**Description**	**Signals**
POV4_solenoid_valve	Opens to supply pneumatic air to actuator	fluid, pneumatic4, unexpected_open4, pneumatic→pneumatic4
POV4_actuator	Rotates ball valve	fluid, pneumatic4, unexpected_open4
POV4_ball_valve:POV4_leak	Valve seat or assembly doesn't seal	fluid, leak
POV4_ball_valve:POV4_stuck	Valve doesn't rotate either open or closed	fluid, pneumatic4, unexpected_open4
POV4_ball_valve:POV4_intermediate	Valve is between open and closed positions	fluid, inter4
ZSC-4	Close limit switch	(none)
ZSO-4	Open limit switch	(none)
Test Point 1: POV-4_zsc		
Tests	**Description**	**Signals**
POV4C_close_cmd	Close limit switch "closed" after valve close command	inter4, unexpected_open4
POV4C_open_cmd	Close limit switch "notClosed" after valve open command	pneumatic4
Test Point 2: POV-4_zso		
Tests	**Description**	**Signals**
POV4O_close_cmd	Open limit switch "notOpen" after valve open command	unexpected_open4
POV4O_open_cmd	Open limit switch "open" after valve open command	inter4, pneumatic4

Table 1: TEAMS shut-off valve component and tests.

4.2 L2

The functions of the solenoid valve, actuator, and ball valve are grouped into one component in the L2 model. The POV-4 component has an open/close command input and a *feedline* (a structure that includes pressure and temperature variables) input that represents the flow of oxygen into the valve. A *feedline* output is the flow out of the valve and two limit switches report the valve status. A valve position attribute is defined to simplify the propositional statements in the nominal and failure modes, which refer only to the valve position. The constraints that apply for the three valve positions are given in Table 2.

The component has two nominal modes, open and closed. When the valve is closed the open limit switch should report "not open" and the closed limit switch should report "closed". If the valve has been closed for a long time we expect the pressure downstream of the valve to be ambient; however, if the valve has recently closed the downstream pressure could be higher than ambient but should be dropping. Therefore, in the logic for the closed mode we state that either the

pressure level out is ambient or the pressure rate is dropping. This is done so that a diagnosis can be done relatively soon after the close command rather than waiting for the pressure to reach the ambient level, which could take several seconds of a short run. When the valve is open the open limit switch should report "open" and the closed limit switch should report "not closed". In this mode we set the input and output to be equal. This makes an assumption that the valve is not choked, which is true except for a short time immediately after opening the valve. The open command will transition the mode from closed to open while the close command will transition the mode from open to closed (assuming the transitions are nominal).

Four fault modes are included in the model and will be considered if the constraints in the intended nominal mode are not satisfied. Three of the fault modes are for a valve that is stuck in an open, intermediate, or closed position.

POV-4 is a fast-acting valve and opens and closes in roughly half a second. After a command is issued to the valve, a diagnosis would not be requested until the valve has completely transitioned so that the constraints in the intended mode are satisfied. Since pressures take more time to stabilize than limit switches, their values may be unassigned before a diagnosis to prevent conflicts or the constraints in the mode may be coded to allow for the observed behavior, as was done for the closed mode. The policy to request diagnoses and unassign transient observations is implemented in Real Time Interface (RTI) code external to the model.

Inputs	Valve command (open, close)		
	Feedline [pressure, temperature]		
Outputs	Open/not open limit switch status		
	Closed/not closed limit switch status		
	Feedline		
Always true	Pressure level out is qualitatively equal to or less than pressure level in		
Attributes	Valve position	Closed (not open & closed)	Pressure level out is ambient OR pressure rate out is dropping
		Intermediate (not open & not closed)	If pressure level in is not ambient, pressure level out is not ambient
		Open (open & not closed)	*Feedline* in and out are equal
Nominal Modes	Open	Valve position is open	
	Closed	Valve position is closed	
Failure Modes	Stuck open	Valve position is open	
	Stuck intermediate	Valve position is intermediate	
	Stuck closed	Valve position is closed	
	Unknown fault	No restrictions	

Table 2: Livingstone valve component.

4.3 RODON

The model of the shutoff valve consists of an assembly of four components—one component corresponds to the ball valve, another represents the function of the solenoid valve and actuators to open or close the valve, and the last two are the closed and open limit switches on the ball valve.

The equations relating pressure drop to mass flow through the valve are taken from Instrumentation, Systems, and Automation (ISA) standards and are shown in Figure 6. Note that the flow coefficient of 322 is for a fully open valve. The valve manufacturer was contacted to get the flow coefficients as the valve rotates from fully closed to fully open. A

theoretical estimate of the valve flow coefficient as a function of degrees open is used instead of the constant CV in the valve mass flow equation.

The valve position (degrees open) is set by a valve driver component that connects to the ball valve component. No attempt is made to model the physics of the actuation process. Instead, the rate at which the actuator opens the ball valve is specified to be between a lower bound and an upper bound. Also specified is an interval for the time delay between the solenoid open command and the start of activation. The time delay and actuation rate intervals are chosen to be slightly greater than the experimentally measured valve response and result in a permissible window of valve position versus time, which is computed directly in the component. An alternative approach is to use the average time delay and actuation rates to compute the valve position outside of the RODON model (in the data processing code). A value for the valve degrees open is passed to the model and has an associated tolerance that allows for deviation from the average values used to compute the valve position. In the physical system, the rate of opening and the observed delay are functions of the pneumatic pressure, starting torque, moving torque, and the geometry of the actuator cylinder and piston, among other things.

As the ball valve rotates from 0 to 90 degrees, the close limit switch value changes from 1 to 0 followed by the open limit switch changing from 0 to 1. The contacts on the switches are adjusted so that the indications will occur within a couple degrees of rotation from the close and open hard stops. Errors in time aligning the sensor data, uncertainty in the logged data, and variability in the valve actuation complicate modeling the value of the open and close limit switch based purely on computed valve rotation. Instead, the status of the valve switches are checked only after the intended action is completed. For example, if the valve is commanded to open the indications of the open and close switches are checked after the computed valve rotation angle has reached 90 degrees rather than checking the close limit switch after only a few degrees of rotation. This was necessary to eliminate spurious diagnoses when processing nominal data from HCF firings.

Shutoff valve model: Instrumentation, Systems and Automation (ISA) standards

$$x = \frac{P_1 - P_2}{P_1} \quad \text{limit } x \le x_T = 0.25$$

$$F_k = \frac{k}{1.4}, F_P = 1, Y = 1 - \frac{x}{3F_k x_T}$$

$$C_V = 322$$

$$Q = \frac{1360}{60} C_V F_P P_1 Y \sqrt{\frac{x}{G_g T Z}} \quad [\text{SCFM}]$$

$$\dot{m} = \frac{1.33}{60 \times 3.28^3} Q \quad [\text{kg/sec}]$$

Constraints:

$\dot{m}_{in} = \dot{m}_{out}$ (Continuity)

$T_{t_{in}} = T_{t_{out}}$ (Adiabatic)

If valve is closed, $\dot{m} = 0$

Figure 6: RODON valve component equations.

5. MODEL COMPARISON

The scope and completeness of the models created with the different tools in this initial study varied. The TEAMS model has the greatest scope, including all of the subsystems of the HCF, but the test code that is necessary to abstract the sensor data to the binary test results that TEAMS uses in the fault diagnosis was not developed. The L2 model has a reduced scope, but the data is abstracted somewhat automatically using a spreadsheet application. However, the

necessary function of the RTI was implemented manually. The RODON model has the smallest scope but is the most complete as far as being able to use the logged data directly to produce a diagnosis.

The TEAMS, L2, and RODON models closely resemble the physical structure of the HCF. Links connecting the components propagate information that may correspond to fundamental measurements such as voltages, currents, pressures, temperatures, and switch positions, or derived quantities such as pressure rate of change. In each of the models, the components capture the functional behavior of the modeled devices. In TEAMS, the functions affected by a component's failure are highly abstracted to a list of signals, or attributes. The signals relate the failure causes (the components) to the manifestations of the failure effects (the tests). While nominal behavior is not modeled explicitly, the device is working properly if all of the tests defined at the test points pass. In L2, the nominal behavior of the component is described by propositional formulae that relate, or constrain, the input and output qualitative values of the components. In RODON, the nominal behavior of the component is specified with quantitative and/or qualitative formulae and logic clauses. One can define failure modes in each of the tools. Real-valued sensor data is abstracted to binary pass/fail test results in TEAMS and to a finite number of bins in L2. RODON does not require such an abstraction. System transients are dealt with externally to the model in TEAMS and L2 while RODON can handle them within the model. TEAMS (with TEAMS-RT) diagnoses the system using pre-compiled fault dictionaries. L2 uses conflicts to focus the diagnostic search procedure to find the most probable combination of components' modes that fits the observations. RODON suspends the constraints of each component and rechecks the model for consistency. Table 3 summarizes the tool comparison.

	TEAMS	**L2**	**RODON**
Behavior description	Via signals in multi-signal flow graph (describes fault propagation)	Via propositional logic	Via logic clauses and numerical equations with interval arithmetic
Abstraction	To binary pass/fail	To discrete bins	Not necessary
Nominal behavior	In test points	In nominal modes	In nominal description
Transients	External	External	Internal or external
Diagnostic strategy	Pre-compiled fault dictionary	Conflict directed best-first search	Constraint suspension

Table 3: Tool comparison.

6. CONCLUSIONS

Models of the HCF were created using three model-based reasoning tools in order to examine their approaches to fault diagnosis of hybrid rocket technologies. The models varied from strictly qualitative (TEAMS and L2) to quantitative (RODON). For the qualitative models, the abstraction code needed for an end-to-end implementation of the fault diagnostic system was not developed. Although rigorous timing studies were not performed, the quantitative model takes much longer to process the data and diagnose the system than the qualitative models, even after considering the additional time that would be required by the abstraction code. The quantitative model offers a relatively straightforward way to handle the system transients by using time as a variable directly in the model. In the qualitative models, more thought must be given to what is expected as the firing progresses and how to capture those expectations in the model and tests (or bins). In general, more knowledge of the system operation must be included outside of the model as the model becomes more abstract.

For the HCF, the introduction of model-based reasoning tools would add little value. The current control system provides effective system safing by monitoring the valve states, burst disks, Programmable Logic Controller (PLC), over and/or under pressures on a couple of pressure measurements, and by having an emergency shutdown button. The HCF operators can do troubleshooting very quickly by examining a few pressure measurements together with the controller alarms and by examining the facility after a firing. The HCF is a topologically linear system with few subsystem interactions, which makes the troubleshooting easier. Furthermore, some of the interactions occur for short durations of time that are scheduled. The pre-defined sequence of events, limited subsystem interactions, expert knowledge of the

operators, and the ability to perform visual inspections after the firing make the addition of an IVHM system superfluous for fault diagnosis of the HCF.

Once the technologies being tested at the HCF are incorporated into a vehicle, the case for an IVHM system becomes stronger. We expect that there will be many more system interactions that are not scheduled. Fault isolation becomes much harder for a human to do efficiently as the system becomes more complex with more measurements and interactions. In addition, we must rely on on-board sensors for diagnostic information. The IVHM system would have to be tightly integrated with a reconfiguration manager or intelligent controller that would continue to meet the vehicle's objectives in a degraded state.

7. FUTURE WORK

In this initial study, we did not have sufficient time to compare the diagnostic capabilities of the model-based reasoning tools. While this is a function of the scope of the models, we want to discern whether or not the more qualitative tools will provide adequate diagnosability of the system. Another tool that was developed in the study but not discussed in this paper is an inductive monitoring system that utilizes techniques from the fields of model-based reasoning, machine learning, and data mining to build system monitoring knowledge bases from archived sensor data. We have begun to investigate how this technique and other algorithms may be used to perform feature extractions that aid the model-based reasoning tools or to perform stand-alone fault detection and identification.

REFERENCES

1. G. Zilliac, M. A. Karabeyoglu, B. Cantwell, R. Hunt, S. DeZilwa, M. Shoffstal, P. Soderman, "Ames Hybrid Combustion Facility", NASA TM 211864, to be published.

2. M. A. Karabeyoglu, B. J. Cantwell, D. Altman, "Development and Testing of Parraffin-based Hybrid Rocket Fuels", *AIAA/ASME/SAE/ASEE 37th Joint Propulsion Conference and Exhibit*, AIAA Paper No. 2001-4503, July 2001.

3. R. Davis, W. Hamscher, "Model-based reasoning: Troubleshooting", *Readings in Model-based Diagnosis*, pp.3-24, editors: W. Hamscher, L. Console, J. de Kleer, Morgan Kaufmann Publishers, Inc., 1992.

4. S. Deb, K. R. Pattipati, V. Raghavan, M. Shakeri, R. Shrestha, "Multi-Signal Flow Graphs: A Novel Approach for System Testability Analysis and Fault Diagnosis", *Proceedings of IEEE AUTOTESTCON*, Anaheim, CA, pp. 361-373, Sept. 1994.

5. B. C. Williams, P. P. Nayak, "A Model-based Approach to Reactive Self-Configuring Systems", *Proceedings of AAAI-96*, pp. 971-978, 1996.

6. J. Kurien, P. Nayak, "Back to the Future for Consistency-based Trajectory Tracking", *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'2000)*, pp. 370-377, 2000.

7. N. Muscettola, P. Nayak, B. Pell, B. Williams, "Remote Agent: To Boldly Go Where No AI System Has Gone Before", *Artificial Intelligence*, **103(1-2)**, pp. 5-47, 1998.

Characterization of Model-based Reasoning Strategies for use in IVHM Architectures

Scott Poll, David Iverson, and Ann Patterson-Hine

Open architectures are gaining popularity for Integrated Vehicle Health Management (IVHM) applications due to the diversity of subsystem health monitoring strategies in use and the need to integrate a variety of techniques at the system health management level. The basic concept of an open architecture suggests that whatever monitoring or reasoning strategy a subsystem wishes to deploy, the system architecture will support the needs of that subsystem and will be capable of transmitting subsystem health status across subsystem boundaries and up to the system level for system-wide fault identification and diagnosis. There is a need to understand the capabilities of various reasoning engines and how they, coupled with intelligent monitoring techniques, can support fault detection and system level fault management.

Researchers in IVHM at NASA Ames Research Center are supporting the development of an IVHM system for liquefying-fuel hybrid rockets. In the initial stage of this project, several readily available reasoning engines were studied to assess candidate technologies for application in next generation launch systems. Hybrid rocket engine experts were readily available for consultation at Ames, thus easing one of the most difficult and time-consuming aspects of building IVHM systems, knowledge acquisition for model construction. Three tools representing the spectrum of model-based reasoning approaches, from a quantitative, simulation based approach to a graph-based, fault propagation technique, were applied to model the behavior of the Hybrid Combustion Facility testbed at Ames. This paper summarizes the characterization of the modeling process for each of the techniques and compares the resulting diagnostic capabilities of the reasoning engines which utilize these models.

Scott Poll is the Project Lead for the Hybrid Combustion Facility IVHM software team at NASA Ames Research Center. He has a B.S. in Aerospace Engineering from the University of Michigan in 1994 and a Master's in Aeronautical Engineering from Caltech in 1995. Recent projects include work in automated reasoning for the X-37 IVHM experiment and in support of the Honeywell IVHM project under NASA's Space Launch Initiative program.